oude appendix

MUNIX

PROGRAMMER'S MANUAL

VOLUME 1

APPENDIX

PCS GmbH
Periphere Computer Systeme
Pfälzer-Wald-Straße 36
8000 München 90
Telefon (0 89) 68 10 21

NAME

as – assembler

SYNOPSIS

as file ...

DESCRIPTION

As assembles the named files, which must have the suffix '.s'. The output of the assembly is left on the corresponding files with suffix '.o'. Assembler listings are produced on the corresponding files with suffix '.lst'.

FILES

/lib/cpp	C preprocessor
/lib/asm	assembler
/usr/tmp/asm*	temporary files
/lib/symfile	used for initialization of /lib/asm
name.s	assembler source
name.o	assembled module
name.lst	assembler listing

SEE ALSO

Ulrike Weng-Beckmann, *Assembler 68000 Users Guide*
Motorola, *Macro Assembler Reference Manual*

DIAGNOSTICS

Diagnostics are given in the assembler listing.

—4 Changes the integer size from 2 (default) to 4. This makes programs which neglect the differences between int and pointer or int and long much more portable. On the other hand, the produced code is less efficient. When the name lcc is linked to cc, lcc is equivalent to cc -4. This option is also passed to the linker.

—2 Like option -4. However, short parameters remain short and char parameters are only converted to short, not int, before they are pushed on the stack. This option is used for example to compile the system calls in the 4-byte-integer standard library /lib/libLc.a, which must interface to a 2-byte-integer world.

—8 Currently, only the Motorola Fast Floating Point Package is available for floating point simulation. This package does not implement 64-bit floating point. Therefore, double is considered to be the same as float by default. Option -8 changes the default such that sizeof(double) = 8. The code generator will then produce calls to procedures dadd, dmul etc. for double addition and multiplication, which are not yet implemented.

—P Run only the macro preprocessor and place the result for each '.c' file in a corresponding '.i' file that has no '#' lines in it.

—E Run only the macro preprocessor and send the result to the standard output. The output is intended for compiler debugging; it is unacceptable as input to *cc*.

—D*name=def*
—D*name*
Define the *name* to the preprocessor, as if by '#define'. If no definition is given, the name is defined as 1.

—U*name*
Remove any initial definition of *name*.

—I*dir* '#include' files whose names do not begin with '/' are always sought first in the directory of the *file* argument, then in directories named in —I options, then in directories on a standard list.

—B*string*
Find substitute compiler passes in the files named *string* with the suffixes cpp, c0, c1 and c2. If *string* is empty, use a standard backup version.

—t[p012]
Find only the designated compiler passes in the files whose names are constructed by a —B option. In the absence of a —B option, the *string* is taken to be '/usr/src/cmd/c/'.

Other arguments are taken to be either linker option arguments, or C-compatible object programs, typically produced by an earlier cc run, or perhaps libraries of C-compatible routines. These programs, together with the results of any compilations specified, are linked (in the order given) to produce an executable program with name **a.out**.

FILES

file.c	input file
file.o	object file
a.out	linked output
/tmp/ctm?	temporaries for *cc*

NAME

cc, lcc – C compiler

SYNOPSIS

cc [option] ... file ...

DESCRIPTION

Cc is the UNIX C compiler. It accepts several types of arguments:

Arguments whose names end with '.c' are taken to be C source programs; they are compiled, and each object program is left on the file whose name is that of the source with '.o' substituted for '.c'. The '.o' file is normally deleted, however, if a single C program is compiled and linked all at one go. This C-compiler has the identifier "m68000" predefined, such that "#ifdef m68000" is true.

A call "cc test.c" is the same as "cc -c test.c && ld /lib/crt0.o test.o -lc && rm test.o". The call "cc *.o" is the same as "ld /lib/crt0.o *.o -lc". If your directory contains the file xyz.c, a call "make xyz" will result in "cc -o xyz xyz.c".

The following options are interpreted by *cc*. See *ld*(1) for link-time options.

–c Suppress the linking phase of the compilation, and force an object file to be produced even if only one program is compiled.

–o *output*
Name the final output file *output*. If this option is used the file 'a.out' will be left undisturbed. This option is passed on to the linker.

–f This option must be given, if your program includes floating point operations. The library libffp.a will be searched before libc.a.

–S Compile the named C programs, and produce an assembly listing on corresponding files suffixed '.s'. This option is used mainly for compiler debugging.

–a This option, followed by a number, e.g. 1234 (decimal), 01234 (octal) or 0x1234 (hex), together with option –S starts the program counter in the produced assembly-listing with the given number.

–L Arrange for the compiler to produce code which puts the current linenumber into the stackframe during execution. The C-stacktrace of *adb*(1) (command $c or $C) will then give you for each active procedure the current linenumber.

–g Adds entries to the symbol table of the produced .o module which indicate line numbers and corresponding code location. Used by the pbug source oriented debugger.

–p Arrange for the compiler to produce code which counts the number of times each routine is called; also, if linking takes place, replace the standard startup routine by one which automatically calls *monitor*(3) at the start and arranges to write out a *mon.out* file at normal termination of execution of the object program. An execution profile can then be generated by use of *prof*(1).

–T With this option initialized data is put into the text- rather than the data segment. Thus for programs like the shell, whose text segments are shared between many users, memory space can be saved. Apply this option only to modules with nothing but constant data declarations.

–u All characters will be treated as "unsigned char".

NAME

f77 – Fortran 77 compiler

SYNOPSIS

f77 [option] ... file ...

DESCRIPTION

F77 is the UNIX Fortran 77 compiler. It accepts several types of arguments:

Arguments whose names end with '.f' are taken to be Fortran 77 source programs; they are compiled, and each object program is left on the file in the current directory whose name is that of the source with '.o' substituted for '.f'.

Arguments whose names end with '.r' or '.e' are taken to be Ratfor or EFL source programs, respectively; these are first transformed by the appropriate preprocessor, then compiled by f77.

In the same way, arguments whose names end with '.c' or '.s' are taken to be C or assembly source programs and are compiled or assembled, producing a '.o' file.

Floating point is implemented with the Motorola Fast Floating Point Package. The compiler will accept double precision declarations, but handles them silently as single precision.

The following options have the same meaning as in *cc*(1). See *ld*(1) for load-time options.

–c Suppress loading and produce '.o' files for each source file.

–p Prepare object files for profiling, see *prof*(1).

–S Compile the named programs, and leave the assembler-listing on corresponding files suffixed '.s'.

–o output
Name the final output file *output* instead of 'a.out'.

The following options are peculiar to *f77*.

–onetrip
Compile DO loops that are performed at least once if reached. (Fortran 77 DO loops are not performed at all if the upper limit is smaller than the lower limit.)

–1 Same as '-onetrip'.

–U Do not convert UPPERCASE to lowercase.

–u Make the default type of a variable 'undefined' rather than using the default Fortran rules.

–C Compile code to check that subscripts are within declared array bounds.

–w Suppress all warning messages. If the option is '–w66', only Fortran 66 compatibility warnings are suppressed.

–Nx Change default sizes of certain tables. 'x' is one of the letters q,x,s,c or n followed directly by a decimal number. The defaults are:

-Nq150	Maximum number of equivalence statements
-Nx200	Maximum number of external definitions
-Ns701	Maximum number of statement labels
-Nc20	Maximum nesting of control structures
-Nn401	Size of hash table

-I2 Use short integers (2 bytes) for Fortran type integer.

-I4 Use long integers (4 bytes) for Fortran type integer (default). Additionally, make subscripts long.

-Is Make subscripts short (default).

-F Apply EFL and Ratfor preprocessor to relevant files, put the result in the file with the suffix changed to '.f', but do not compile.

-m Apply the M4 preprocessor to each '.r' or '.e' file before transforming it with the Ratfor or EFL preprocessor.

-Ex Use the string *x* as an EFL option in processing '.e' files.

-Rx Use the string *x* as a Ratfor option in processing '.r' files.

Other arguments are taken to be either loader option arguments, or F77-compatible object programs, typically produced by an earlier run, or perhaps libraries of F77-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with name 'a.out'.

FILES

file.[fresc]	input file
file.o	object file
a.out	loaded output
/usr/lib/f77pass1	compiler
/lib/c1	pass 2
/lib/c2	pass3
/usr/lib/libF77.a	intrinsic function library
/usr/lib/libI77.a	Fortran I/O library
/lib/libc.a	C library, see section 3
/lib/libm.a	Mathematical library
/lib/libfp.a	Floating point library

SEE ALSO

S. I. Feldman, P. J. Weinberger, *A Portable Fortran 77 Compiler*
prof(1), cc(1), ld(1)

DIAGNOSTICS

The diagnostics produced by *f77* itself are intended to be self-explanatory.. Occasional messages may be produced by the loader.

BUGS

This compiler has run the Fortran Compiler Validation Suite FCVS 78 from the Federal Cobol Compiler Test Service (FCCTS). FCVS 78 is also known as "navytest". It passed all tests except two. One error is due to floating point accuracy. The other error is that a format statement must be given before an "assign"-statement that assigns the label of the format statement to an integer variable.

/lib/cpp	preprocessor
/lib/c[012]	compiler passes for *cc*
/usr/c/oc[012]	backup compiler passes for *cc*
/usr/c/ocpp	backup preprocessor
/lib/crt0.o	runtime initialization
/lib/mcrt0.o	runtime initialization for profiling
/lib/libc.a	standard library, see *intro*(3)
/lib/libfp.a	library with floating point routines
/usr/include	standard directory for '#include' files

SEE ALSO

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978
D. M. Ritchie, *C Reference Manual*
monitor(3), prof(1), adb(1), ld(1)

DIAGNOSTICS

The diagnostics produced by C itself are intended to be self-explanatory. Occasional messages may be produced by the linker.

BUGS

Option -g should also be usable by adb, so that breakpoints could be set to the start of a line.

NAME

cpio − copy file archives in and out

SYNOPSIS

cpio −o [acBv]

cpio −i [Bcdmrtuv6] [patterns]

cpio −p [adlmruv] directory

DESCRIPTION

Cpio −o (copy out) reads the standard input to obtain a list of path names and copies those files onto the standard output together with path name and status information.

Cpio −i (copy in) extracts from the standard input (which is assumed to be the product of a previous **cpio −o**) the names of files selected by zero or more *patterns* given in the name-generating notation of *sh*(1). In *patterns*, meta-characters ?, *, and [...] match the slash / character. The default for *patterns* is * (i.e., select all files).

Cpio −p (pass) copies out and in in a single operation. Destination path names are interpreted relative to the named *directory*.

The meanings of the available options are:

- **a** Reset access times of input files after they have been copied.
- **B** Input/output is to be blocked 5,120 bytes to the record (does not apply to the *pass* option; meaningful only with data directed to or from **/dev/rmt?**).
- **d** *Directories* are to be created as needed.
- **c** Write *header* information in ASCII character form for portability.
- **r** Interactively *rename* files. If the user types a null line, the file is skipped.
- **t** Print a *table of contents* of the input. No files are created.
- **u** Copy *unconditionally* (normally, an older file will not replace a newer file with the same name).
- **v** *Verbose*: causes a list of file names to be printed. When used with the **t** option, the table of contents looks like the output of an **ls −l** command (see *ls*(1)).
- **l** Whenever possible, link files rather than copying them. Usable only with the **−p** option.
- **m** Retain previous file modification time. This option is ineffective on directories that are being copied.
- **6** Process an old (i.e., UNIX *Sixth* Edition format) file. Only useful with **−i** (copy in).

EXAMPLES

The first example below copies the contents of a directory into an archive; the second duplicates a directory hierarchy:

```
ls | cpio −o >/dev/mt0

cd olddir
find . −print | cpio −pdl newdir
```

The trivial case "**find . −print | cpio −oB >/dev/rmt0**" can be handled more efficiently by:

```
find . −cpio /dev/rmt0
```

SEE ALSO

ar(1), find(1), cpio(5).

BUGS

Path names are restricted to 128 characters. If there are too many unique linked files, the program runs out of memory to keep track of them and, thereafter, linking information is lost. Only the super-user can copy special files.

NAME

devnm – device name

SYNOPSIS

/etc/devnm [names]

DESCRIPTION

Devnm identifies the special file associated with the mounted file system where the argument *name* resides.

This command is most commonly used by /etc/rc (see *rc*(8)) to construct a mount table entry for the root device.

EXAMPLE

The command:

```
/etc/devnm /usr
```

produces

```
rp1 /usr
```

if /usr is mounted on /dev/rp1.

FILES

/dev/rp*
/etc/mtab

SEE ALSO

setmnt(1M).

NAME

f77 – Fortran 77 compiler

SYNOPSIS

f77 [option] ... file ...

DESCRIPTION

F77 is the UNIX Fortran 77 compiler. It accepts several types of arguments:

Arguments whose names end with '.f' are taken to be Fortran 77 source programs; they are compiled, and each object program is left on the file in the current directory whose name is that of the source with '.o' substituted for '.f'.

Arguments whose names end with '.r' or '.e' are taken to be Ratfor or EFL source programs, respectively; these are first transformed by the appropriate preprocessor, then compiled by f77.

In the same way, arguments whose names end with '.c' or '.s' are taken to be C or assembly source programs and are compiled or assembled, producing a '.o' file.

Floating point is implemented with the Motorola Fast Floating Point Package. The compiler will accept double precision declarations, but handles them silently as single precision.

The following options have the same meaning as in *cc*(1). See *ld*(1) for load-time options.

–c Suppress loading and produce '.o' files for each source file.

–p Prepare object files for profiling, see *prof*(1).

–S Compile the named programs, and leave the assembler-listing on corresponding files suffixed '.s'.

–o output
Name the final output file *output* instead of 'a.out'.

The following options are peculiar to *f77*.

–onetrip
Compile DO loops that are performed at least once if reached. (Fortran 77 DO loops are not performed at all if the upper limit is smaller than the lower limit.)

–1 Same as '-onetrip'.

–U Do not convert UPPERCASE to lowercase.

–u Make the default type of a variable 'undefined' rather than using the default Fortran rules.

–C Compile code to check that subscripts are within declared array bounds.

–w Suppress all warning messages. If the option is '–w66', only Fortran 66 compatibility warnings are suppressed.

–Nx Change default sizes of certain tables. 'x' is one of the letters q,x,s,c or n followed directly by a decimal number. The defaults are:

-Nq150	Maximum number of equivalence statements
-Nx200	Maximum number of external definitions
-Ns701	Maximum number of statement labels
-Nc20	Maximum nesting of control structures
-Nn401	Size of hash table

—I2 Use short integers (2 bytes) for Fortran type integer.

—I4 Use long integers (4 bytes) for Fortran type integer (default). Additionally, make subscripts long.

—Is Make subscripts short (default).

—F Apply EFL and Ratfor preprocessor to relevant files, put the result in the file with the suffix changed to '.f', but do not compile.

—m Apply the M4 preprocessor to each '.r' or '.e' file before transforming it with the Ratfor or EFL preprocessor.

—E*x* Use the string *x* as an EFL option in processing '.e' files.

—R*x* Use the string *x* as a Ratfor option in processing '.r' files.

Other arguments are taken to be either loader option arguments, or F77-compatible object programs, typically produced by an earlier run, or perhaps libraries of F77-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with name 'a.out'.

FILES

file.[fresc]	input file
file.o	object file
a.out	loaded output
/usr/lib/f77pass1	compiler
/lib/c1	pass 2
/lib/c2	pass3
/usr/lib/libF77.a	intrinsic function library
/usr/lib/libI77.a	Fortran I/O library
/lib/libc.a	C library, see section 3
/lib/libm.a	Mathematical library
/lib/libffp.a	Floating point library

SEE ALSO

S. I. Feldman, P. J. Weinberger, *A Portable Fortran 77 Compiler*
prof(1), cc(1), ld(1)

DIAGNOSTICS

The diagnostics produced by *f77* itself are intended to be self-explanatory.. Occasional messages may be produced by the loader.

BUGS

This compiler has run the Fortran Compiler Validation Suite FCVS 78 from the Federal Cobol Compiler Test Service (FCCTS). FCVS 78 is also known as "navytest". It passed all tests except two. One error is due to floating point accuracy. The other error is that a format statement must be given before an "assign"-statement that assigns the label of the format statement to an integer variable.

NAME

ld – loader

SYNOPSIS

ld [option] file ...

DESCRIPTION

Ld combines several object programs into one, resolves external references, and searches libraries. In the simplest case several object *files* are given, and *ld* combines them, producing an object module which can be either executed or become the input for a further *ld* run. (In the latter case, the **–r** option must be given to preserve the relocation bits.) The output of *ld* is left on **a.out**. This file is made executable only if no errors occurred during the load.

The argument routines are concatenated in the order specified. The entry point of the output is the beginning of the first routine.

If any argument is a library, it is searched exactly once at the point it is encountered in the argument list. Only those routines defining an unresolved external reference are loaded. If a routine from a library references another routine in the library, and the library has not been processed by *ranlib*(1), the referenced routine must appear after the referencing routine in the library. Thus the order of programs within libraries may be important. If the first member of a library is named '__.SYMDEF', then it is understood to be a dictionary for the library such as produced by *ranlib*; the dictionary is searched iteratively to satisfy as many references as possible.

The symbols '_etext', '_edata' and '_end' ('etext', 'edata' and 'end' in C) are reserved, and if referred to, are set to the first location above the program, the first location above initialized data, and the first location above all data respectively. It is erroneous to define these symbols.

Ld understands several options. Except for **–l**, they should appear before the file names.

–s 'Strip' the output, that is, remove the symbol table and relocation bits to save space (but impair the usefulness of the debugger). This information can also be removed by *strip*(1).

–u Take the following argument as a symbol and enter it as undefined in the symbol table. This is useful for loading wholly from a library, since initially the symbol table is empty and an unresolved reference is needed to force the loading of the first routine.

–l*x* This option is an abbreviation for the library name '/lib/lib*x*.a', where *x* is a string. If that does not exist, *ld* tries '/usr/lib/lib*x*.a'. A library is searched when its name is encountered, so the placement of a **–l** is significant.

–4 This option changes the library name '/lib/lib*x*.a' to '/lib/libL*x*.a'. For example instead of /lib/libc.a the library /lib/libLc.a is searched. These libraries are assumed to include modules which have been produced with cc -4.

–x Do not preserve local (non-.globl) symbols in the output symbol table; only enter external symbols. This option saves some space in the output file.

–r Generate relocation bits in the output file so that it can be the subject of another *ld* run. This flag also prevents final definitions from being given

NAME

install − install commands

SYNOPSIS

install [−c dira] [−f dirb] [−i] [−n dirc] [−o] [−s] file [dirx ...]

DESCRIPTION

Install is a command most commonly used in "makefiles" (see *make*(1)) to install a *file* (updated target file) in a specific place within a file system. Each *file* is installed by copying it into the appropriate directory, thereby retaining the mode and owner of the original command. The program prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* ...) are given, *install* will search (using *find*(1)) a set of default directories (/bin, /usr/bin, /etc, /lib, and /usr/lib, in that order) for a file with the same name as *file*. When the first occurrence is found, *install* issues a message saying that it is overwriting that file with *file*, and proceeds to do so. If the file is not found, the program states this and exits without further action.

If one or more directories (*dirx* ...) are specified after *file*, those directories will be searched before the directories specified in the default list.

The meanings of the options are:

−c *dira* Installs a new command in the directory specified in *dira*. Looks for *file* in *dira* and installs it there if it is not found. If it is found, *install* issues a message saying that the file already exists, and exits without overwriting it. May be used alone or with the **−s** option.

−f *dirb* Forces *file* to be installed in given directory, whether or not one already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to 755 and bin, respectively. If the file already exists, the mode and owner will be that of the already existing file. May be used alone or with the **−o** or **−s** options.

−i Ignores default directory list, searching only through the given directories (*dirx* ...). May be used alone or with any other options other than **−c** and **−f**.

−n *dirc* If *file* is not found in any of the searched directories, it it put in the directory specified in *dirc*. The mode and owner of the new file will be set to 755 and bin, respectively. May be used alone or with any other options other than **−c** and **−f**.

−o If *file* is found, this option saves the "found" file by copying it to OLD*file* in the directory in which it was found. May be used alone or with any other options other than **−c**.

−s Suppresses printing of messages other than error messages. May be used alone or with any other options.

−F cause directories to be marked with a trailing '/' and executable files to be marked with a trailing '•'; this is the default if the last character of the name the program is invoked with is a 'f'.

−R recursively list subdirectories encountered.

The mode printed under the −l option contains 11 characters which are interpreted as follows: the first character is

d if the entry is a directory;
b if the entry is a block-type special file;
c if the entry is a character-type special file;
m if the entry is a multiplexor-type character special file;
− if the entry is a plain file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next to permissions to others in the same user-group; and the last to all others. Within each set the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, 'execute' permission is interpreted to mean permission to search the directory for a specified file. The permissions are indicated as follows:

r if the file is readable;
w if the file is writable;
x if the file is executable;
− if the indicated permission is not granted.

The group-execute permission character is given as s if the file has set-group-ID mode; likewise the user-execute permission character is given as s if the file has set-user-ID mode.

The last character of the mode (normally 'x' or '−') is t if the 1000 bit of the mode is on. See *chmod*(1) for the meaning of this mode.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks is printed.

FILES

/etc/passwd to get user ID's for 'ls −l'.
/etc/group to get group ID's for 'ls −g'.

BUGS

Newline and tab are considered printing characters in file names.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as "ls −s" is much different than "ls −s | lpr". On the other hand, not doing this setting would make old shell scripts which used *ls* almost certain losers.

Column widths choices are poor for terminals which can tab.

NAME

ls − list contents of directory

SYNOPSIS

ls [−abcdfgilmqrstuxlCFR] name ...
l [*ls* options] name ...

DESCRIPTION

For each directory argument, *ls* lists the contents of the directory; for each file argument, *ls* repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The format chosen depends on whether the output is going to a teletype, and may also be controlled by option flags. The default format for a teletype is to list the contents of directories in multi-column format, with the entries sorted down the columns. (Files which are not the contents of a directory being interpreted are always sorted across the page rather than down the page in columns. This is because the individual file names may be arbitrarily long.) If the standard output is not a teletype, the default format is to list one entry per line. Finally, there is a stream output format in which files are listed across the page, separated by ',' characters. The −m flag enables this format; when invoked as *l* this format is also used.

There are an unbelievable number of options:

−l List in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file. (See below.) If the file is a special file the size field will instead contain the major and minor device numbers.

−t Sort by time modified (latest first) instead of by name, as is normal.

−a List all entries; usually '.' and '..' are suppressed.

−s Give size in blocks, including indirect blocks, for each entry.

−d If argument is a directory, list only its name, not its contents (mostly used with −l to get status on directory).

−r Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.

−u Use time of last access instead of last modification for sorting (−t) or printing (−l).

−c Use time of file creation for sorting or printing.

−i Print i-number in first column of the report for each file listed.

−f Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off −l, −t, −s, and −r, and turns on −a; the order is the order in which entries appear in the directory.

−g Give group ID instead of owner ID in long listing.

−m force stream output format

−1 force one entry per line output format, e.g. to a teletype

−C force multi-column output, e.g. to a file or a pipe

−q force printing of non-graphic characters in file names as the character '?'; this normally happens only if the output device is a teletype

−b force printing of non-graphic characters to be in the \ddd notation, in octal.

−x force columnar printing to be sorted across rather than down the page; this is the default if the last character of the name the program is invoked with is an 'x'.

to common symbols, and suppresses the 'undefined symbol' diagnostics.

-d Force definition of common storage even if the **-r** flag is present.

-o The *name* argument after **-o** is used as the name of the *ld* output file, instead of **a.out**.

-p This argument produces a symbol listing on stdout.

-y This argument links objects for an unmapped address range, such that code, data and bss are contiguous, starting from 0. Useful for Macsbug etc.

-b This option is used to link unix or standalone programs, where code is in segment 0 and data and bss in segment 1. If neither **-b** nor **-y** are given, normal user-mode programs loadable by the kernel are produced, with code in segment 6 and data and bss in segment 8.

-m This option produces "S-records" for Macsbug on file dm.out.

-e The following argument is taken to be the name of the entry point of the loaded program; location 0 is the default.

-O This is an overlay file, only the text segment will be replaced by *exec*(2). Shared data must have the same layout as in the program overlaid.

-D The next argument is a decimal number that sets the size of the data segment.

FILES

/lib/lib*.a	libraries
/lib/libI.*.a	libraries with four-byte integers
/usr/lib/lib*.a	more libraries
a.out	output file

SEE ALSO

as(1), ar(1), cc(1), ranlib(1)

BUGS

Overlays are not tested.

NAME

newconf – generate configuration file and reconfigure MUNIX

SYNOPSIS

/etc/newconf [configuration file]

DESCRIPTION

Newconf makes a new MUNIX kernel from an already existing or from an interactively created configuration file.

If no configuration file is specified *newconf* asks for

– the device drivers to be included
(tty driver is automatically included)

– assignation of DMA extension registers to DMA devices

– the type and unit of the root and swap device

– the origin (block number) and size of the swap area

– some other system parameters

Newconf creates the configuration file */usr/sys/conf.h* as include file to the configuration table */usr/sys/c.c* and the interrupt vector table */usr/sys/l.s*.

If a configuration file is specified this file is copied to */usr/sys/conf.h*.

Reconfiguration of MUNIX is automatically done by compiling */usr/sys/c.c*, assembling */usr/sys/l.s* and linking */usr/sys/c.o*, */usr/sys/l.o*, the kernel library */usr/sys/lib1* and the driver library */usr/sys/lib2* to a new MUNIX kernel named */nunix*.

For later use you should safe the interactively created configuration file */usr/sys/conf.h* by renaming it.

FILES

/usr/sys/conf.h
/usr/sys/c.*
/usr/sys/l.*
/usr/sys/lib1
/usr/sys/lib2
/nunix

SEE ALSO

whatconf (1m)

BUGS

NAME

pack - packs or unpacks many files <---> one.

SYNOPSIS

a) pack file1 [file2] ... [filen] or
b) pack

DESCRIPTION

a) pack file1 ... filen
packs these files onto stdout. Between the files the following line is inserted

(/***************<file 1>******************/)

b) pack
unpacks stdin (a previously packed file) onto the component files.
Hint: fgrep '^(/' packedfile
picks out the names of the component files).

EXAMPLE1

pack *.p | xref
makes a cross-reference list of a group of files.

EXAMPLE2

pack *.c | sed -f change.sed | pack
seds a whole group of files at once.

DIAGNOSTICS

If file cannot be opened the message

'cannot open file <file>' is written on stderr.

If the inputfile has a bad format the message
'phase error in line <line number>' is written on stderr.

NAME

rxctrl – floppy disk manipulating program

SYNOPSIS

rxctrl – [fsdiz] /dev/rrx?

DESCRIPTION

Rxctrl is used to give commands to the floppy drive:

f Initialize data fields according to specified floppy density. See RX(4).
s Byte swapping
d Default: no swap of bytes

Unusual commands:

i Interleave of sectors off.
z Zigzag on: sequential blocks on different floppy sides.

FILES

/dev/rrx*

SEE ALSO

rx(4), format(8), tmctrl(1), dd(1)

NAME

tmctrl – magnetic tape manipulating program

SYNOPSIS

tmctrl [**–s**] [**–d**] [**tapename**]

DESCRIPTION

Tmctrl is used to give commands to the tape drive. If no tapename is specified, /dev/rmt0 is used.

Here are the commands:

-s byte swapping
-d default: no swap of bytes

FILES

/dev/rmt* /dev/nrmt* Raw magnetic tape interface

SEE ALSO

tm(4), rxctrl(1), dd(1)

NAME

tplot − graphics filters

SYNOPSIS

tplot [−Tterminal [−e raster]]

DESCRIPTION

These commands read plotting instructions (see *plot*(5)) from the standard input and in general produce, on the standard output, plotting instructions suitable for a particular *terminal*. If no *terminal* is specified, the environment parameter **$TERM** (see *environ*(7)) is used. Known *terminal*s are:

300	DASI 300.
300S	DASI 300s.
450	DASI 450.
4014	Tektronix 4014.
ver	Versatec D1200A. This version of *plot* places a scan-converted image in **/usr/tmp/raster$$** and sends the result directly to the plotter device, rather than to the standard output. The −e option causes a previously scan-converted file *raster* to be sent to the plotter.

FILES

/usr/lib/t300
/usr/lib/t300s
/usr/lib/t450
/usr/lib/t4014
/usr/lib/vplot
/usr/tmp/raster$$

SEE ALSO

plot(3X), plot(5)

NAME

vfontinfo – inspect and print out information about unix fonts

SYNOPSIS

vfontinfo [**–v**] fontname [characters]

DESCRIPTION

Vfontinfo allows you to examine a font in the unix format. It prints out all the information in the font header and information about every non-null (width > 0) glyph. This can be used to make sure the font is consistent with the format.

The *fontname* argument is the name of the font you wish to inspect. It writes to standard output. If it can't find the file in your working directory, it looks in /usr/lib/vfont (the place most of the fonts are kept).

The *characters*, if given, specify certain characters to show. If omitted, the entire font is shown.

If the **–v** (verbose) flag is used, the bits of the glyph itself are shown as an array of X's and spaces, in addition to the header information.

SEE ALSO

vpr(1), vfont(5)
The Berkeley Font Catalog

AUTHORS

Mark Horton
Andy Hertzfeld

NAME

volcopy, labelit – copy file systems with label checking

SYNOPSIS

/etc/volcopy [**–bpi***bits-per-inch*] [**–feet***size*] *fsname special1 volname1 special2 volname2*

/etc/labelit *special* [*fsname volume* [**–n**]]

DESCRIPTION

Volcopy makes a literal copy of the file system using a blocksize matched to the device (10 blocks for 800/1600 bpi tape; 88 blocks for everything else). Using *volcopy*, a 2400 foot/1600 bpi tape will hold a 65K file system. The optional flag arguments are used only with tapes (**–bpi** -- bits-per-inch; **–feet** -- size of reel in feet). The program requests the information if it is not given on the command line. If the file system is too large to fit on one reel, *volcopy* will prompt for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two drives.

The *fsname* argument represents the mounted name (e.g.: **root**, **u1**, etc.) of the filsystem being copied.

The *special* should be the physical disk section or tape (e.g.: **/dev/rrp15**, **/dev/rmt0**, etc.).

The *volname* is the physical volume name (e.g.: **pk3**, **t0122**, etc.) and should match the external label sticker. Such label names are limited to five or fewer characters.

Special1 and *volname1* are the device and volume from which the copy of the file system is being extracted. *Special2* and *volname2* are the target device and volume.

Fsname and *volname* are recorded in the last 12 characters of the superblock (char fsname[6], volname[6];).

Labelit can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, *labelit* prints current label values. The **–n** option provides for initial labeling of new tapes only (this destroys previous contents).

FILES

/etc/log/filesave a record of file systems/volumes copied

SEE ALSO

filsys (5)

BUGS

Only device names beginning **/dev/rmt** are treated as tapes.

NAME

vpr, vprm, vpq, vprint − raster printer/plotter spooler

SYNOPSIS

vpr [−W] [−l] [−v] [−t [−1234 font]] [−w] [−w*width*] [−m] [name ...]
vprm [id ...] [filename ...] [owner ...]
vpq
vprint [−W] file ...

DESCRIPTION

Vpr causes the named files to be queued for printing or typeset simulation on one of the available raster printer/plotters. If no files are named, the standard input is read. By default the input is assumed to be line printer-like text. For very wide plotters, the input is run through the filter */usr/lib/sidebyside* giving it an argument of −w106 which arranges it four pages adjacent with 90 column lines (the rest is for the left margin). Since there are 8 lines per inch in the default printer font, *vpr* thus produces 86 lines per page (the top and bottom lines are left blank).

The following options are available:

−l — Print the input in a more literal manner. Page breaks are not inserted, and most control characters (except format effectors: \n, \f, etc.) are printed (many control characters print special graphics not in the ASCII character set.) Tab and underline processing is still done. If this option is not given, control characters which are not format effectors are ignored, and page breaks are inserted after an appropriate number of lines have been printed on a page.

−W — Queues files for printing on a wide output device, if available. Normally, files are queued for printing on a narrow output device.

−1234 — Specifies a font to be mounted on font position *i*. The daemon will construct a *.railmag* file referencing */usr/lib/vfont/name.size*.

−m — Report by *mail*(1) when printing is complete.

−w — (Applicable only to wide output devices.) Do not run the input through sidebyside. Such processing has been done already, or full (440 character) printer width is desired.

−w*width* — Use width *width* rather than 90 for *sidebyside*.

−v — Use the filter */usr/lib/vrast* to convert the vectors to raster. The named files must be a parameter and vector file (in that order) created by *versaplot*(3x) routines.

−t — Use the filter */usr/lib/vcat* to typeset the input on the printer/plotter. The input must have been generated by *troff*(1) run with the −t option. This is not normally run directly to wide output devices, since it is wasteful to run only one page across. The program *vtroff*(1) is normally used and arranges, using *vsort*(1) for printing to occur four pages across, conserving paper.

Vprm removes entries from the raster device queues. The id, filename or owner should be that reported by *vpq*. All apropriate files will be removed. Both queues are always searched. The id of each file removed from the queue will be printed.

Vpq prints the queues. Each entry in the queue is printed showing the owner of the queue entry, an identification number, the size of the entry in characters, and the file which is to be printed. The *id* is useful for removing a specific entry from the printer queue using *vprm*

Vprint is a shell script which *pr's* a copy of each named file on one of the electrostatic printer/plotters. The files are normally printed on a narrow device; −W option causes them to be printed on a wide device.

FILES

/usr/spool/v?d/*	device spool areas
/usr/lib/v?d	daemons
/usr/lib/vpd	Versatec daemon
/usr/lib/vpf	filter for printer simulation
/usr/lib/*vcat	filter for typeset simulation
/usr/lib/vrast	filter for versaplot
/usr/lib/sidebyside	filter for wide output

SEE ALSO

troff(1), vfont(5), vp(4), pti(1), vtroff(1), versaplot(3x)

BUGS

You can't run bit maps in a queued fashion to the plotters. This is because the volume of the data (more than 1 Megabyte per vertical foot) is unwieldy. Instead you must follow the instructions in *vp*(4) and run your program when the plotter is idle, or run it in the background and have it wait for the device to become idle.

Queued jobs print in directory (seemingly random) order. The plotters are fast enough that this is not a real problem.

The 1's (one's) and l's (lower-case el's) in a Benson-Varian's standard character set look very similar; caution is advised.

Vprm should have options allowing just one of the queues to be searched.

A versatec's hardware character set is rather ugly. *Vprint* should use one of the constant width fonts to produce prettier listings.

NAME

vtroff – troff to a raster plotter

SYNOPSIS

vtroff [**−w**] [**−F** majorfont] [**−123** minorfont] [**−l***length*] [**−x**] troff arguments

DESCRIPTION

Vtroff runs *troff*(1) sending its output through various programs to produce typeset output on a raster plotter such as a Benson-Varian or or a Versatec. The **−W** option specifies that a wide output device be used; the default is to use a narrow device. The **−l** (lower case l) option causes the output to be split onto successive pages every *length* inches rather than the default 11".

The default font is a Hershey font. If some other font is desired you can give a **−F** argument and then the font name. This will place normal, italic and bold versions of the font on positions 1, 2, and 3. To place a font only on a single position, you can give an argument of the form **−***n* and the minor font name. A *.r* will be added to the minor font name if needed. Thus "vtroff −ms paper" will set a paper in the Hershey font, while "vtroff −F nonie −ms paper" will set the paper in the (sans serif) nonie font. The **−x** option asks for exact simulation of photo-typesetter output. (I.e. using the width tables for the C.A.T. photo-typesetter)

FILES

/usr/lib/tmac/tmac.vcat	default font mounts and bug fixes
/usr/lib/fontinfo/*	fixes for other fonts
/usr/lib/vfont	directory containing fonts

SEE ALSO

troff(1), nettroff(1), vfont(5), vpr(1)

BUGS

Since some macro packages work correctly only if the fonts named R, I, B, and S are mounted, and since the Versatec fonts have different widths for individual characters than the fonts found on the typesetter, the following dodge was necessary: If you don't use the ".fp" troff directive then you get the widths of the standard typesetter fonts suitable for shipping the output of troff over the network to the computer center A machine for phototypesetting. If, however, you remount the R, I, B and S fonts, then you get the width tables for the Versatec.

NAME

whatconf – what device drivers are in an unix kernel

SYNOPSIS

/etc/whatconf unixkernel

DESCRIPTION

Whatconf tells you what devices the specified unix kernel is configured for.

SEE ALSO

newconf(1m)

NAME

xd - hexadecimal dump

SYNOPSIS

xd [file] [offset1[.][b] [- [offset2[.][b]]

DESCRIPTION

Xd dumps file in hexadecimal and as characters.

The file argument specifies which file is to be dumped. If no file argument is specified, the standard input is used.

The offset argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as hexadecimal bytes. If it starts with '0', the offset is interpreted in octal. If '.' is appended, the offset is interpreted in decimal. If 'b' is appended, the offset is interpreted in blocks of 512 bytes. If the file argument is omitted, the offset argument must be preceded '+'.

If no second offset is specified, dumping continues until end-of-file. Otherwise dumping ends before the specified endaddress. The interpretation of offset2 is the same as above.

SEE ALSO

adb(1)

NAME

tgetent, tgetnum, tgetflag, tgetstr, tgoto, tputs — terminal independent operation routines

SYNOPSIS

```
char PC;
char *BC;
char *UP;
short ospeed;

tgetent(bp, name)
char *bp, *name;

tgetnum(id)
char *id;

tgetflag(id)
char *id;

char *
tgetstr(id, area)
char *id, **area;

char *
tgoto(cm, destcol, destline)
char *cm;

tputs(cp, affcnt, outc)
register char *cp;
int affcnt;
int (*outc)();
```

DESCRIPTION

These functions extract and use capabilities from the terminal capability data base *termcap*(5). These are low level routines; see *curses*(3) for a higher level package.

Tgetent extracts the entry for terminal *name* into the buffer at *bp*. *Bp* should be a character buffer of size 1024 and must be retained through all subsequent calls to *tgetnum*, *tgetflag*, and *tgetstr*. *Tgetent* returns −1 if it cannot open the *termcap* file, 0 if the terminal name given does not have an entry, and 1 if all goes well. It will look in the environment for a TERMCAP variable. If found, and the value does not begin with a slash, and the terminal type name is the same as the environment string TERM, the TERMCAP string is used instead of reading the termcap file. If it does begin with a slash, the string is used as a path name rather than */etc/termcap*. This can speed up entry into programs that call *tgetent*, as well as to help debug new terminal descriptions or to make one for your terminal if you can't write the file */etc/termcap*.

Tgetnum gets the numeric value of capability *id*, returning −1 if is not given for the terminal. *Tgetflag* returns 1 if the specified capability is present in the terminal's entry, 0 if it is not. *Tgetstr* gets the string value of capability *id*, placing it in the buffer at *area*, advancing the *area* pointer. It decodes the abbreviations for this field described in *termcap*(5), except for cursor addressing and padding information.

Tgoto returns a cursor addressing string decoded from *cm* to go to column *destcol* in line *destline*. It uses the external variables UP (from the up capability) and BC (if bc is given rather than bs) if necessary to avoid placing \n, ^D or ^@ in the returned string. (Programs which call tgoto should be sure to turn off the XTABS bit(s), since tgoto may now output a tab. Note that programs using termcap should in general turn off XTABS anyway since some terminals use control I for other functions, such as nondestructive space.) If a % sequence is given which is not understood, then *tgoto* returns "OOPS".

Tputs decodes the leading padding information of the string *cp*; *affcnt* gives the number of lines affected by the operation, or 1 if this is not applicable, *outc* is a routine which is called with each character in turn. The external variable *ospeed* should contain the output speed of the terminal as encoded by *stty (2)*. The external variable PC should contain a pad character to be used (from the pc capability) if a null (^@) is inappropriate.

FILES

/usr/lib/libtermcap.a	−ltermcap library
/etc/termcap	data base

SEE ALSO

ex(1), curses(3), termcap(5)

AUTHOR

William Joy

BUGS

NAME

hk – RK611/RK06, RK07 moving-head disk

DESCRIPTION

The octal representation of the minor device number is encoded *dp*, where *d* is a physical drive number, and *p* is a logical unit (subsection) within a physical unit. The origins and sizes of the logical units on each drive, counted in cylinders of 66 512-byte blocks, are:

log. unit	start cyl.	length	size (in blocks)	
0	0	146	9636	(root on RK06/07)
1	146	135	8910	(swap on RK06/07)
2	281	129	8514	(rest of RK06)
3	281	533	35178	(rest of RK07)
4	0	0	0	(spare)
5	0	0	0	(spare)
6	0	410	27060	(all of RK06)
7	0	814	53724	(all of RK07)

Systems distributed for these devices use disk 0 for the root, disk 1 for swapping, and disk 6 (RK06) or disk 7 (RK07) for a mounted user file system.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records.

A "raw' interface provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.' In raw I/O the buffer must begin on a word boundary.

FILES

/dev/hk?, /dev/rhk?

SEE ALSO

format (8)

BUGS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*, *write* and *lseek*(2) should always deal in 512-byte multiples.

NAME

hp – RP04/05/06, RM02/03 moving-head disk

DESCRIPTION

The octal representation of the minor device number is encoded *dp*, where *d* is a physical drive number, and *p* is a logical unit (subsection) within a physical unit. The origins and sizes of the logical units on each drive, counted in cylinders of 418 512-byte blocks, for the RP04/05/06 are:

log. unit	start cyl.	length	size (in blocks)	
0	0	23	9614	(root on RP04/05/06)
1	23	21	8778	(swap on RP04/05/06)
2	44	21	8778	(/sys on RP04/05/06)
3	65	345	144210	(rest of RP04/05)
4	65	749	313082	(rest of RP06)
5	411	403	168454	(/usr on RP06)
6	0	410	171380	(all of RP04/05)
7	0	814	340252	(all of RP06)

The logical units for the RM02/03 are:

log. unit	start cyl.	length	size (in blocks)	
0	0	60	9600	(root on RM02/03)
1	60	55	8800	(swap on RM02/03)
2	115	50	8000	(/sys on RM02/03)
3	165	657	105120	(rest of RM02/03)
4	0	0	0	(spare)
5	0	0	0	(spare)
6	0	0	0	(spare)
7	0	822	131520	(all of RM02/03)

Systems distributed for these devices use disk 0 for the root, disk 1 for swapping, and disk 3 (RP04/05, RM02/03) or disk 4 (RP06) for a mounted user file system.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records.

A 'raw' interface provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.' In raw I/O the buffer must begin on a word boundary.

FILES

/dev/rp?, /dev/rrp?
/dev/rm?, /dev/rrm?

SEE ALSO

format (8)

BUGS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*, *write* and *lseek*(2) should always deal in 512-byte multiples.

NAME

rx – RX01 or RX02 floppy disk

DESCRIPTION

The *rx* driver supports both double sided (DS) or single sided (SS) drives and double density (DD) or single density (SD) format.
Rx? refers to an entire disk as a single sequentially-addressed file. Its 256-word blocks are numbered 0 to (number of blocks – 1). For the number of blocks on a floppy disk see the following table:

500	SS/SD
1000	DS/SD
1001	SS/DD
2002	DS/DD

Rx0 refer to drive 0, single density format, *rx1* to drive 1, single density format, *rx2* to drive 0, double density format and *rx3* refer to drive 1, double density format.

The *rx* files discussed above access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a 'raw' interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw RX files begin with *rrx*.

In raw I/O the buffer must begin on a word boundary, and counts should be a multiple of 512 bytes (a disk block).

FILES

/dev/rx?, /dev/rrx?

SEE ALSO

rxctrl (1), format (8)

BUGS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*, *write* and *lseek*(2) should always deal in 512-byte multiples.

NAME

st – Streamer interface

DESCRIPTION

The files *st0*, *rst0*, *nst0*, *nrst0* refer to the Archive streamer. The letter r indicates "raw" device, the letter n indicates "no rewind" when the streamer is closed. For nst0 and nrst0 the bit 0200 in the minor device number must be set.

If the tape does not rewind, then the action depends on the open mode. When the streamer is opened for read, the tape will skip forward to the next tapemark. When the streamer was opened for write, a tapemark will be written and the tape stops after the tapemark. Thus, to skip to the next tapemark, you can give the command " < /dev/nst0 "; to rewind, you can say " < /dev/st0 ".

As usual, the block devices st0 and nst0 only allow the transfer of 512-byte blocks. The raw devices rst0 and nrst0 allow transfers of up to and including 127 512-byte blocks with a single read or write. However, the byte count must always be an exact multiple of 512.

Unfortunately, when copying blocks from disk to tape, the streamer needs data faster than Unix can provide. Therefore, the streamer over- or underruns and does not stream. This means that after it has transferred some data, the tape will stop. When subsequent data arrives, it will rewind a piece of tape, then read forward to where it stopped, and immediately write the data. This zigzag-motion of the tape can be very time-consuming. The "zig" occurs, when it transfers data, and its time is proportional to the amount of data transferred. The "zag" is the rewind, and its time is constant. The ratio of "zig"-time to "zag"-time becomes tolerable, when the amount of data transferred with one "zig" gets large.

Two programs have been modified to work with larger buffersizes when working with the streamer: cpio and volcopy. For both programs, the option –S sets the blocking factor to high values (127 for cpio, 110 for volcopy). The program volcopy now also exists in a standalone version. These programs allow you to backup your disks properly. Once in a while you should make a physical copy of your root file system with volcopy. The standalone version of volcopy can then be used to recreate a root file system after a catastrophic failure. At least once a day you should make a total or incremental dump of all files with cpio. From the cpio-tapes you can easily retrieve single files or whole directories.

EXAMPLES

total dump:

```
find / -print | cpio -oS >/dev/rst0
```

incremental dump (last three days, say)

```
find / -mtime -3 -print | cpio -oS >/dev/rst0
```

file retrieval:

```
cpio -ivS myfile </dev/rst0
```

physical dump:

```
labelit /dev/rst0 root tape1 -n
volcopy -S root /dev/rhk0 hk0 /dev/rst0 tape1
```

standalone (only terminal input shown):

```
/sa/volcopy
g
-S root st(0,0) tape1 hk(0,0) root
```

SEE ALSO

volcopy(1), cpio(1), labelit(1)

BUGS

While rewinding to Begin Of Tape (BOT) the system pauses. Sometimes, when a program operating on the streamer is interrupted, the system blocks.

NAME

fstab – static information about the filesystems

SYNOPSIS

#include <fstab.h>

DESCRIPTION

The file */etc/fstab* contains descriptive information about the various file systems. */etc/fstab* is only *read* by programs, and not written; it is the duty of the system administrator to properly create and maintain this file.

These programs use */etc/fstab: dump, mount, umount, swapon, fsck* and *df*. The order of records in */etc/fstab* is important, for both *fsck*, *mount*, and *umount* sequentially iterate through */etc/fstab* doing their thing.

The special file name is the block special file name, and not the character special file name. If a program needs the character special file name, the program must create it by appending a "r" after the last "/" in the special file name.

If *fs_type* is "rw" or "ro" then the file system whose name is given in the *fs_file* field is normally mounted read-write or read-only on the specified special file. The *fs_freq* field is used for these file systems by the *dump*(8) command to determine which file systems need to be dumped. The *fs_passno* field is used by the *fsck*(8) program to determine the order in which file system checks are done at reboot time. The root file system should be specified with a *fs_passno* of 1, and other file systems should have larger numbers. File systems within a drive should have distinct numbers, but file systems on different drives can be checked on the same pass to utilize parallelism available in the hardware.

If *fs_type* is "sw" then the special file is made avaiable as a piece of swap space by the *swapon*(8) command at the end of the system reboot procedure. The fields other than *fs_spec* and *fs_type* are not used in this case.

Fs_type may be specified as "xx" to cause an entry to be ignored. This is useful to show disk partitions which are currently not used but will be used later.

```
#define FSTAB                   "/etc/fstab"
#define FSNMLG                  16

#define FSTABFMT    "%16s:%16s:%2s:%d:%d\n"
#define FSTABARG(p) (p)->fs_spec, (p)->fs_file, \
                    (p)->fs_type, &(p)->fs_freq, &(p)->fs_passno
#define FSTABNARGS 5

#define FSTAB_RW    "rw"    /* read write device */
#define FSTAB_RO    "ro"    /* read only device */
#define FSTAB_SW    "sw"    /* swap device */
#define FSTAB_XX    "xx"    /* ignore totally */

struct fstab {
        char    fs_spec[FSNMLG]; /* block special device name */
        char    fs_file[FSNMLG]; /* file system path prefix */
        char    fs_type[3];      /* rw,ro,sw or xx */
        int     fs_freq;         /* dump frequency, in days */
        int     fs_passno;       /* pass number on parallel dump */
};
```

The proper way to read records from */etc/fstab* is to use the routines getfsent(), getfsspec() or getfsfile().

FILES

/etc/fstab

SEE ALSO

getfsent(3)

NAME

termcap – terminal capability data base

SYNOPSIS

/etc/termcap

DESCRIPTION

Termcap is a data base describing terminals, used, *e.g.*, by *vi*(1) and *curses*(3). Terminals are described in *termcap* by giving a set of capabilities which they have, and by describing how operations are performed. Padding requirements and initialization sequences are included in *termcap*.

Entries in *termcap* consist of a number of ':' separated fields. The first entry for each terminal gives the names which are known for the terminal, separated by '|' characters. The first name is always 2 characters long and is used by older version 6 systems which store the terminal type in a 16 bit word in a systemwide data base. The second name given is the most common abbreviation for the terminal, and the last name given should be a long name fully identifying the terminal. The second name should contain no blanks; the last name may well contain blanks for readability.

CAPABILITIES

(P) indicates padding may be specified
(P•) indicates that padding may be based on no. lines affected

Name	Type	Pad?	Description
ae	str	(P)	End alternate character set
al	str	(P•)	Add new blank line
am	bool		Terminal has automatic margins
as	str	(P)	Start alternate character set
bc	str		Backspace if not ^H
bs	bool		Terminal can backspace with ^H
bt	str	(P)	Back tab
bw	bool		Backspace wraps from column 0 to last column
CC	str		Command character in prototype if terminal settable
cd	str	(P•)	Clear to end of display
ce	str	(P)	Clear to end of line
ch	str	(P)	Like cm but horizontal motion only, line stays same
cl	str	(P•)	Clear screen
cm	str	(P)	Cursor motion
co	num		Number of columns in a line
cr	str	(P•)	Carriage return, (default ^M)
cs	str	(P)	Change scrolling region (vt100), like cm
cv	str	(P)	Like ch but vertical only.
da	bool		Display may be retained above
dB	num		Number of millisec of bs delay needed
db	bool		Display may be retained below
dC	num		Number of millisec of cr delay needed
dc	str	(P•)	Delete character
dF	num		Number of millisec of ff delay needed
dl	str	(P•)	Delete line
dm	str		Delete mode (enter)
dN	num		Number of millisec of nl delay needed
do	str		Down one line
dT	num		Number of millisec of tab delay needed
ed	str		End delete mode

ei	str		End insert mode; give ":ei=:" if ic
eo	str		Can erase overstrikes with a blank
ff	str	(P*)	Hardcopy terminal page eject (default ^L)
hc	bool		Hardcopy terminal
hd	str		Half-line down (forward 1/2 linefeed)
ho	str		Home cursor (if no cm)
hu	str		Half-line up (reverse 1/2 linefeed)
hz	str		Hazeltine; can't print ~'s
ic	str	(P)	Insert character
if	str		Name of file containing is
im	bool		Insert mode (enter); give ":im=:" if ic
in	bool		Insert mode distinguishes nulls on display
ip	str	(P*)	Insert pad after character inserted
is	str		Terminal initialization string
k0-k9	str		Sent by "other" function keys 0-9
kb	str		Sent by backspace key
kd	str		Sent by terminal down arrow key
ke	str		Out of "keypad transmit" mode
kh	str		Sent by home key
kl	str		Sent by terminal left arrow key
kn	num		Number of "other" keys
ko	str		Termcap entries for other non-function keys
kr	str		Sent by terminal right arrow key
ks	str		Put terminal in "keypad transmit" mode
ku	str		Sent by terminal up arrow key
l0-l9	str		Labels on "other" function keys
li	num		Number of lines on screen or page
ll	str		Last line, first column (if no cm)
ma	str		Arrow key map, used by vi version 2 only
mi	bool		Safe to move while in insert mode
ml	str		Memory lock on above cursor.
mu	str		Memory unlock (turn off memory lock).
nc	bool		No correctly working carriage return (DM2500,H2000)
nd	str		Non-destructive space (cursor right)
nl	str	(P*)	Newline character (default \n)
ns	bool		Terminal is a CRT but doesn't scroll.
os	bool		Terminal overstrikes
pc	str		Pad character (rather than null)
pt	bool		Has hardware tabs (may need to be set with is)
se	str		End stand out mode
sf	str	(P)	Scroll forwards
sg	num		Number of blank chars left by so or se
so	str		Begin stand out mode
sr	str	(P)	Scroll reverse (backwards)
ta	str	(P)	Tab (other than ^I or with padding)
tc	str		Entry of similar terminal - must be last
te	str		String to end programs that use cm
ti	str		String to begin programs that use cm
uc	str		Underscore one char and move past it
ue	str		End underscore mode
ug	num		Number of blank chars left by us or ue
ul	bool		Terminal underlines even though it doesn't overstrike

up	str	Upline (cursor up)
us	str	Start underscore mode
vb	str	Visible bell (may not move cursor)
ve	str	Sequence to end open/visual mode
vs	str	Sequence to start open/visual mode
xb	bool	Beehive (f1=escape, f2=ctrl C)
xn	bool	A newline is ignored after a wrap (Concept)
xr	bool	Return acts like ce \r \n (Delta Data)
xs	bool	Standout not erased by writing over it (HP 264?)
xt	bool	Tabs are destructive, magic so char (Teleray 1061)

A Sample Entry

The following entry, which describes the Concept-100, is among the more complex entries in the *termcap* file as of this writing. (This particular concept entry is outdated, and is used as an example only.)

```
c1|c100|concept100:is=\EU\Ef\E7\E5\E8\El\ENH\EK\E\200\Eo&\200:\
        :al=3*\E^R:am:bs:cd=16*\E^C:ce=16\E^S:cl=2*^L:cm=\Ea%+ %+ :co#80:\
        :dc=16\E^A:dl=3*\E^B:ei=\E\200:eo:im=\E^P:in:ip=16*:li#24:mi:nd=\E=:\
        :se=\Ed\Ee:so=\ED\EE:ta=8\t:ul:up=\E;:vb=\Ek\EK:xn:
```

Entries may continue onto multiple lines by giving a \ as the last character of a line, and that empty fields may be included for readability (here between the last field on a line and the first field on the next). Capabilities in *termcap* are of three types: Boolean capabilities which indicate that the terminal has some particular feature, numeric capabilities giving the size of the terminal or the size of particular delays, and string capabilities, which give a sequence which can be used to perform particular terminal operations.

Types of Capabilities

All capabilities have two letter codes. For instance, the fact that the Concept has "automatic margins" (i.e. an automatic return and linefeed when the end of a line is reached) is indicated by the capability am. Hence the description of the Concept includes am. Numeric capabilities are followed by the character '#' and then the value. Thus co which indicates the number of columns the terminal has gives the value '80' for the Concept.

Finally, string valued capabilities, such as ce (clear to end of line sequence) are given by the two character code, an '=', and then a string ending at the next following ':'. A delay in milliseconds may appear after the '=' in such a capability, and padding characters are supplied by the editor after the remainder of the string is sent to provide this delay. The delay can be either a integer, e.g. '20', or an integer followed by an '*', i.e. '3*'. A '*' indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. When a '*' is specified, it is sometimes useful to give a delay of the form '3.5' specify a delay per unit to tenths of milliseconds.

A number of escape sequences are provided in the string valued capabilities for easy encoding of characters there. A \E maps to an ESCAPE character, ^x maps to a control-x for any appropriate x, and the sequences \n \r \t \b \f give a newline, return, tab, backspace and formfeed. Finally, characters may be given as three octal digits after a \, and the characters ^ and \ may be given as \^ and \\. If it is necessary to place a : in a capability it must be escaped in octal as \072. If it is necessary to place a null character in a string capability it must be encoded as \200. The routines which deal with *termcap* use C strings, and strip the high bits of the output very late so that a \200 comes out as a \000 would.

Preparing Descriptions

We now outline how to prepare descriptions of terminals. The most effective way to prepare a terminal description is by imitating the description of a similar terminal in *termcap* and to build up a description gradually, using partial descriptions with *ex* to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the *termcap* file to describe it or bugs in *ex*. To easily test a new terminal description you can set the environment variable TERMCAP to a pathname of a file containing the description you are working on and the editor will look there rather than in */etc/termcap*. TERMCAP can also be set to the termcap entry itself to avoid reading the file when starting up the editor. (This only works on version 7 systems.)

Basic capabilities

The number of columns on each line for the terminal is given by the co numeric capability. If the terminal is a CRT, then the number of lines on the screen is given by the li capability. If the terminal wraps around to the beginning of the next line when it reaches the right margin, then it should have the am capability. If the terminal can clear its screen, then this is given by the cl string capability. If the terminal can backspace, then it should have the bs capability, unless a backspace is accomplished by a character other than ^H (ugh) in which case you should give this character as the bc string capability. If it overstrikes (rather than clearing a position when a character is struck over) then it should have the os capability.

A very important point here is that the local cursor motions encoded in *termcap* are undefined at the left and top edges of a CRT terminal. The editor will never attempt to backspace around the left edge, nor will it attempt to go up locally off the top. The editor assumes that feeding off the bottom of the screen will cause the screen to scroll up, and the am capability tells whether the cursor sticks at the right edge of the screen. If the terminal has switch selectable automatic margins, the *termcap* file usually assumes that this is on, i.e. am.

These capabilities suffice to describe hardcopy and "glass-tty" terminals. Thus the model 33 teletype is described as

```
t3|33|tty33:co#72:os
```

while the Lear Siegler ADM-3 is described as

```
cl|adm3|3|si adm3:am:bs:cl=^Z:li#24:co#80
```

Cursor addressing

Cursor addressing in the terminal is described by a cm string capability, with *printf*(3s) like escapes %x in it. These substitute to encodings of the current line or column position, while other characters are passed through unchanged. If the cm string is thought of as being a function, then its arguments are the line and then the column to which motion is desired, and the % encodings have the following meanings:

%d	as in *printf*, 0 origin
%2	like %2d
%3	like %3d
%.	like %c
%+x	adds *x* to value, then %.
%>xy	if value > x adds y, no output.
%r	reverses order of line and column, no output
%i	increments line/column (for 1 origin)
%%	gives a single %
%n	exclusive or row and column with 0140 (DM2500)
%B	BCD (16*(x/10)) + (x%10), no output.
%D	Reverse coding (x-2*(x%16)), no output. (Delta Data).

Consider the HP2645, which, to get to row 3 and column 12, needs to be sent \E&a12c03Y padded for 6 milliseconds. Note that the order of the rows and columns is inverted here, and that the row and column are printed as two digits. Thus its cm capability is "cm=6\E&%r%2c%2Y". The Microterm ACT-IV needs the current row and column sent preceded by a ^T, with the row and column simply encoded in binary, "cm=^T%.%.". Terminals which use "%." need to be able to backspace the cursor (bs or bc), and to move the cursor up one line on the screen (up introduced below). This is necessary because it is not always safe to transmit \t, \n ^D and \r, as the system may change or discard them.

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus "cm=\E=%+ %+ ".

Cursor motions

If the terminal can move the cursor one position to the right, leaving the character at the current position unchanged, then this sequence should be given as nd (non-destructive space). If it can move the cursor up a line on the screen in the same column, this should be given as up. If the terminal has no cursor addressing capability, but can home the cursor (to very upper left corner of screen) then this can be given as ho; similarly a fast way of getting to the lower left hand corner can be given as ll; this may involve going up with up from the home position, but the editor will never do this itself (unless ll does) because it makes no assumption about the effect of moving up from the home position.

Area clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as ce. If the terminal can clear from the current position to the end of the display, then this should be given as cd. The editor only uses cd from the first column of a line.

Insert/delete line

If the terminal can open a new blank line before the line where the cursor is, this should be given as al; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as dl; this is done only from the first position on the line to be deleted. If the terminal can scroll the screen backwards, then this can be given as sb, but just al suffices. If the terminal can retain display memory above then the da capability should be given; if display memory can be retained below then db should be given. These let the editor understand that deleting a line on the screen may bring non-blank lines up from below or that scrolling back with sb may bring down non-blank lines.

Insert/delete character

There are two basic kinds of intelligent terminals with respect to insert/delete character which can be described using *termcap*. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can find out which kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type "abc def" using local cursor motions (not spaces) between the "abc" and the "def". Then position the cursor before the "abc" and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the "abc" shifts over to the "def" which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and should give the capability in, which stands for "insert null". If your terminal does something different and unusual then you

may have to modify the editor to get it to use the insert mode your terminal defines. We have seen no terminals which have an insert mode not not falling into one of these two classes.

The editor can handle both terminals which have an insert mode, and terminals which send a simple sequence to open a blank position on the current line. Give as im the sequence to get into insert mode, or give it an empty value if your terminal uses a sequence to insert a blank position. Give as ei the sequence to leave insert mode (give this, with an empty value also if you gave im so). Now give as ic any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give ic, terminals which send a sequence to open a screen position should give it here. (Insert mode is preferable to the sequence to open a position on the screen if your terminal has both.) If post insert padding is needed, give this as a number of milliseconds in ip (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in ip.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (e.g. if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability mi to speed up inserting in this case. Omitting mi will affect only speed. Some terminals (notably Datamedia's) must not have mi because of the way their insert mode works.

Finally, you can specify delete mode by giving dm and ed to enter and exit delete mode, and dc to delete a single character while in delete mode.

Highlighting, underlining, and visible bells

If your terminal has sequences to enter and exit standout mode these can be given as so and se respectively. If there are several flavors of standout mode (such as inverse video, blinking, or underlining — half bright is not usually an acceptable "standout" mode unless the terminal is in inverse video mode constantly) the preferred mode is inverse video by itself. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as the TVI 912 and Teleray 1061 do, this is acceptable, and although it may confuse some programs slightly, it can't be helped.

Codes to begin underlining and end underlining can be given as us and ue respectively. If the terminal has a code to underline the current character and move the cursor one space to the right, such as the Microterm Mime, this can be given as uc. (If the underline code does not move the cursor to the right, give the code followed by a nondestructive space.)

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement) then this can be given as vb; it must not move the cursor. If the terminal should be placed in a different mode during open and visual modes of *ex*, this can be given as vs and ve, sent at the start and end of these modes respectively. These can be used to change, e.g., from a underline to a block cursor and back.

If the terminal needs to be in a special mode when running a program that addresses the cursor, the codes to enter and exit this mode can be given as ti and te. This arises, for example, from terminals like the Concept with more than one page of memory. If the terminal has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the terminal for cursor addressing to work properly.

If your terminal correctly generates underlined characters (with no special codes needed) even though it does not overstrike, then you should give the capability ul. If overstrikes are erasable with a blank, then this should be indicated by giving eo.

Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local (this applies, for example, to the unshifted HP 2621 keys). If the keypad can be set to

transmit or not transmit, give these codes as ks and ke. Otherwise the keypad is assumed to always transmit. The codes sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as kl, kr, ku, kd, and kh respectively. If there are function keys such as f0, f1, ..., f9, the codes they send can be given as k0, k1, ..., k9. If these keys have labels other than the default f0 through f9, the labels can be given as l0, l1, ..., l9. If there are other keys that transmit the same code as the terminal expects for the corresponding function, such as clear screen, the *termcap* 2 letter codes can be given in the ko capability, for example, ":ko=cl,ll,sf,sb:", which says that the terminal has clear, home down, scroll down, and scroll up keys that transmit the same thing as the cl, ll, sf, and sb entries.

The ma entry is also used to indicate arrow keys on terminals which have single character arrow keys. It is obsolete but still in use in version 2 of vi, which must be run on some minicomputers due to memory limitations. This field is redundant with kl, kr, ku, kd, and kh. It consists of groups of two characters. In each group, the first character is what an arrow key sends, the second character is the corresponding vi command. These commands are h for kl, j for kd, k for ku, l for kr, and H for kh. For example, the mime would be :ma=^Kj^Zk^Xl: indicating arrow keys left (^H), down (^K), up (^Z), and right (^X). (There is no home key on the mime.)

Miscellaneous

If the terminal requires other than a null (zero) character as a pad, then this can be given as pc.

If tabs on the terminal require padding, or if the terminal uses a character other than ^I to tab, then this can be given as ta.

Hazeltine terminals, which don't allow "~" characters to be printed should indicate hz. Datamedia terminals, which echo carriage-return linefeed for carriage return and then ignore a following linefeed should indicate nc. Early Concept terminals, which ignore a linefeed immediately after an am wrap, should indicate xn. If an erase-eol is required to get rid of standout (instead of merely writing on top of it), xs should be given. Teleray terminals, where tabs turn all characters moved over to blanks, should indicate xt. Other specific terminal problems may be corrected by adding more capabilities of the form xx.

Other capabilities include is, an initialization string for the terminal, and if, the name of a file containing long initialization strings. These strings are expected to properly clear and then set the tabs on the terminal, if the terminal has settable tabs. If both are given, is will be printed before if. This is useful where if is */usr/lib/tabset/std* but is clears the tabs first.

Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability tc can be given with the name of the similar terminal. This capability must be *last* and the combined length of the two entries must not exceed 1024. Since *termlib* routines search the entry from left to right, and since the tc capability is replaced by the corresponding entry, the capabilities given at the left override the ones in the similar terminal. A capability can be cancelled with xx@ where xx is the capability. For example, the entry

```
hn|2621nl:ks@:ke@:tc=2621:
```

defines a 2621nl that does not have the ks or ke capabilities, and hence does not turn on the function key labels when in visual mode. This is useful for different modes for a terminal, or for different user preferences.

FILES

/etc/termcap file containing terminal descriptions

SEE ALSO

ex(1), curses(3), termcap(3), tset(1), vi(1), ul(1), more(1)

AUTHOR

William Joy
Mark Horton added underlining and keypad support

BUGS

Ex allows only 256 characters for string capabilities, and the routines in *termcap(3)* do not check for overflow of this buffer. The total length of a single entry (excluding only escaped newlines) may not exceed 1024.

The ma, vs, and ve entries are specific to the *vi* program.

Not all programs support all entries. There are entries that are not supported by any program.

NAME

ttytype – data base of terminal types by port

SYNOPSIS

/etc/ttytype

DESCRIPTION

Ttytype is a database containing, for each tty port on the system, the kind of terminal that is attached to it. There is one line per port, containing the terminal kind (as a name listed in termcap (5)), a space, and the name of the tty, minus /dev/.

This information is read by *tset*(1) and by *login*(1) to initialize the TERM variable at login time.

SEE ALSO

tset(1), login(1)

BUGS

Some lines are merely known as "dialup" or "plugboard".

NAME

vfont – font formats for the Benson-Varian or Versatec

SYNOPSIS

/usr/lib/vfont/*

DESCRIPTION

The fonts for the printer/plotters have the following format. Each file contains a header, an array of 256 character description structures, and then the bit maps for the characters themselves. The header has the following format:

```
struct header {
        short           magic;
        unsigned short  size;
        short           maxx;
        short           maxy;
        short           xtnd;
} header;
```

The *magic* number is 0436 (octal). The *maxx*, *maxy*, and *xtnd* fields are not used at the current time. *Maxx* and *maxy* are intended to be the maximum horizontal and vertical size of any glyph in the font, in raster lines. The *size* is the size of the bit maps for the characters in bytes. Before the maps for the characters is an array of 256 structures for each of the possible characters in the font. Each element of the array has the form:

```
struct dispatch {
        unsigned short  addr;
        short           nbytes;
        char            up;
        char            down;
        char            left;
        char            right;
        short           width;
};
```

The *nbytes* field is nonzero for characters which actually exist. For such characters, the *addr* field is an offset into the rest of the file where the data for that character begins. There are *up*+*down* rows of data for each character, each of which has *left*+*right* bits, rounded up to a number of bytes. The *width* field is not used by vcat, although it is used by *vwidth*(1) to make width tables for *troff*. It represents the logical width of the glyph, in raster lines, and shows where the base point of the next glyph would be.

FILES

/usr/lib/vfont/*

SEE ALSO

troff(1), pti(1), vpr(1), vtroff(1), vwidth(1), vfontinfo(1), fed(1)

NAME

fish – play "Go Fish"

SYNOPSIS

/usr/games/fish

DESCRIPTION

Fish plays the game of "Go Fish", a childrens' card game. The Object is to accumulate 'books' of 4 cards with the same face value. The players alternate turns; each turn begins with one player selecting a card from his hand, and asking the other player for all cards of that face value. If the other player has one or more cards of that face value in his hand, he gives them to the first player, and the first player makes another request. Eventually, the first player asks for a card which is not in the second player's hand: he replies 'GO FISH!' The first player then draws a card from the 'pool' of undealt cards. If this is the card he had last requested, he draws again. When a book is made, either through drawing or requesting, the cards are laid down and no further action takes place with that face value.

To play the computer, simply make guesses by typing a, 2, 3, 4, 5, 6, 7, 8, 9, 10, j, q, or k when asked. Hitting return gives you information about the size of my hand and the pool, and tells you about my books. Saying 'p' as a first guess puts you into 'pro' level: The default is pretty dumb.

NAME

hangman – Computer version of the game hangman

SYNOPSIS

/usr/games/hangman

DESCRIPTION

In *hangman*, the computer picks a word from the on-line word list and you must try to guess it. The computer keeps track of which letters have been guessed and how many wrong guesses you have made on the screen in a graphic fashion.

FILES

/usr/dict/words On-line word list

AUTHOR

Modified for terminal graphics from the original source from BTL by Michael Toy.

BUGS

NAME

worm – Play the growing worm game

SYNOPSIS

worm [*size*]

DESCRIPTION

In *worm*, you are a little worm, your body is the "o"'s on the screen and your head is the "@". You move with the hjkl keys (as in the game snake). If you don't press any keys, you continue in the direction you last moved. The upper case HJKL keys move you as if you had pressed several (9 for HL and 5 for JK) of the corrosponding lower case key (unless you run into a digit, then it stops).

On the screen you will see a digit, if your worm eats the digit is will grow longer, the actual amount longer depends on which digit it was that you ate. The object of the game is to see how long you can make the worm grow.

The game ends when the worm runs into either the sides of the screen, or itself. The current score (how much the worm has grown) is kept in the upper left corner of the screen.

The optional argument, if present, is the initial length of the worm.

BUGS

If the initial length of the worm is set to less than one or more than 75, various strange things happen.

NAME

worms — animate worms on a display terminal

SYNOPSIS

worms [-field] [-length #] [-number #] [-trail]

DESCRIPTION

Brian Horn (cithep!bdh) showed me a *TOPS-20* program on the DEC-2136 machine called *WORM*, and suggested that I write a similar program that would run under *Unix*. I did, and no apologies.

-field makes a "field" for the worm(s) to eat; -trail causes each worm to leave a trail behind it. You can figure out the rest by yourself.

FILES

/etc/termcap

AUTHOR

Eric P. Scott

SEE ALSO

Snails, by Karl Heuer

DIAGNOSTICS

Invalid length
: Value not in range 2 <= length <= 1024

Invalid number of worms
: Value not in range 1 <= number <= 40

TERM: parameter not set
: The TERM environment variable is not defined. Do

```
TERM=terminal type
export TERM
```

Unknown terminal type
: Your terminal type (as determined from the TERM environment variable) is not defined in /etc/termcap.

Terminal not capable of cursor motion
: Your terminal is too stupid to run this program.

Out of memory
: This should never happen on a VAX.

BUGS

The lower-right-hand character position will not be updated properly on a terminal that wraps at the right margin.

Terminal initialization is not performed.

There should be an option to have the worms eat *Pink Floyd* lyrics.

NAME

configuration information – table of interrupt vector and device addresses

SYNOPSIS

cat /usr/sys/confinfo

DESCRIPTION

Confinfo is a table of the interrupt vector and device addresses used in UNIX. It contains:

Configuration Information					
Device	Interrupt Vec.		Device Address		max. Units/Lines
	octal	hex	octal	hex	
console	60	C0	777560	FFFF70	1
hb	70	E0	767770	FFEFF8	2 Drives
rl	160	1C0	774400	FFF900	4 Drives
vp	174 204	1F0 plot 210 print	777400	FFFF00	1
lp	200	200	777514	FFFF4C	1
hk	210	220	777440	FFFF20	6 Drives
rk	220	240	777400	FFFF00	4 Drives
tm	224	250	772520	FFF550	8 Drives
hp	254	2B0	776700	FFFDC0	2 Drives
rx2	264	2D0	777170	FFFE78	2 Drives
st	270	2E0	777600 777640	FFFF80 csr FFFFA0 data	1 Drive
tty					7 Lines
1	300	300	776500	FFFD40	
2	310	320	776510	FFFD48	
3	320	340	776520	FFFD50	
4	330	360	776530	FFFD58	
5	340	380	776540	FFFD60	
6	350	3A0	776550	FFFD68	
7	360	3C0	776560	FFFD70	
dz(v)					32(16) Lines
1st	330	360	760100	FFE040	
2nd	340	380	760110	FFE048	
3rd	350	3A0	760120	FFE050	
4th	360	3C0	760130	FFE058	
dh	340	380	760020	FFE010	16 Lines
td	370	3E0	777600 777640	FFFF80 csr FFFFA0 data	2 Drives

FILES

/usr/sys/confinfo

NAME

format – how to format disks

SYNOPSIS

/bin/rxctrl –f
/sa/rxformat
/sa/rlformat
/sa/rmformat
/sa/emuformat
/sa/xyloformat

DESCRIPTION

Rxctrl formats a 5 1/4" floppy emulating a single sided, double density 8" floppy with a ANDROMEDA WDC11 controller. For further details see RXCTRL(1) the floppy driver manipulation program.

All the other formatting programs are standalone programs. Enter the Minitor (Type sync, push INIT) and type i.e.:

```
.rl          (load from RL02)
./sa/emuformat (executable file)
.g0          (start the program)
```

Rxformat formats a 8" floppy RX02 compatible. The floppy controller responses '$'. Type

XD2 (double density) or
XD1 (single density) <cr>
and
XU0 (left drive) or
XU1 (right drive) <cr>

Rlformat formats a whole TANDON TM603SE drive with a ANDROMEDA WDC11 controller as a RL02. The following arguments are interactively asked for:

```
UNIT:        0 (TANDON drive 0)
             1 (TANDON drive 1)
INTERLEAVE: 1..31 (odd)
             Use 5 to optimize the seek time.
```

Rmformat formats one or all tracks of a FUJITSU M2312K drive with a DATARAM SO4/A controller as a RM02. The following arguments are interactively asked for:

```
SINGLE TRACK ? 'y' or <cr>
if yes: HEAD: 0..4
        TRACK: 0..822
```

Emuformat formats a whole FUJITSU M2312K drive with a EMULEX SC02 controller as

```
Unit 0: RK07   Unit 3: RK07
Unit 1: RK07   Unit 4: RK07
Unit 2: RK06   Unit 5: RK06
```

The following argument is interactively asked for:

```
UNIT: 0..5
```

Xyloformat formats one or all tracks of a FUJITSU M2312K drive with a XYLOGICS 550 controller as

```
Unit 0: RK07
Unit 1: RK07
Unit 2: RK06
```

The following arguments are interactively asked for:

```
          UNIT: 0..2
SINGLE TRACK ? 'y' or <cr>
if yes  HEAD: 0..2
        TRACK: 0..410 for RK06
               0..814 for RK07
```

SEE ALSO

rl(4), rx(4), hp(4), hk(4), mkfs(1M)

BUGS

The Minitor response the loading command with 'can't find file'. Ignore the message!

NAME

fsck – file system consistency check and interactive repair

SYNOPSIS

/etc/fsck –p [filesystem ...]
/etc/fsck [**–y**] [**–n**] [**–sX**] [**–SX**] [**–t** filename] [filesystem] ...

DESCRIPTION

The first form of *fsck* preens a standard set of filesystems or the specified file systems. It is normally used in the script /etc/rc during automatic reboot. In this case *fsck* reads the table /etc/fstab to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible. Normally, the root file system will be checked on pass 1, other "root" ("a" partition) file systems on pass 2, other small file systems on separate passes (e.g. the "d" file systems on pass 3 and the "e" file systems on pass 4), and finally the large user file systems on the last pass, e.g. pass 5. A pass number of 0 in fstab causes a disk to not be checked; similarly partitions which are not shown as to be mounted "rw" or "ro" are not checked.

The system takes care that only a restricted class of innocuous inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

Unreferenced inodes

Link counts in inodes too large

Missing blocks in the free list

Blocks in the free list also in files

Counts in the super-block wrong

These are the only inconsistencies which *fsck* with the **–p** option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature of the correction. After successfully correcting a file system, *fsck* will print the number of files on that file system and the number of used and free blocks.

Without the **–p** option, *fsck* audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that a number of the corrective actions which are not fixable under the **–p** option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond yes or no. If the operator does not have write permission *fsck* will default to a **–n** action.

Fsck has more consistency checks than its predecessors *check*, *dcheck*, *fcheck*, and *icheck* combined.

The following flags are interpreted by *fsck*.

–y Assume a yes response to all questions asked by *fsck*; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

–n Assume a no response to all questions asked by *fsck*; do not open the file system for writing.

–s*X* Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-

core copy of the superblock will not continue to be used, or written on the file system.

The −s*X* option allows for creating an optimal free-list organization. The following forms of *X* are supported for the following devices:

−s3 (RP03)
−s4 (RP04, RP05, RP06)
−sBlocks-per-cylinder:Blocks-to-skip (for anything else)

If *X* is not given, the values used when the filesystem was created are used. If these values were not specified, then the value *400:9* is used.

−S*X* Conditionally reconstruct the free list. This option is like −s*X* above except that the free list is rebuilt only if there were no discrepancies discovered in the file system. Using −S will force a no response to all questions asked by *fsck*. This option is useful for forcing free list reorganization on uncontaminated file systems.

−t If *fsck* cannot obtain enough memory to keep its tables, it uses a scratch file. If the −t option is specified, the file named in the next argument is used as the scratch file, if needed. Without the −t flag, *fsck* will prompt the operator for the name of the scratch file. The file chosen should not be on the filesystem being checked, and if it is not a special file or did not already exist, it is removed when *fsck* completes.

If no filesystems are given to *fsck* then a default list of file systems is read from the file /etc/fstab.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   Directory size not 16-byte aligned.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   File pointing to unallocated inode.
   Inode number out of range.
8. Super Block checks:
   More than 65536 inodes.
   More blocks for inodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the lost+found directory. The name assigned is the inode number. The only restriction is that the directory lost+found must preexist in the root of the filesystem being checked and must have empty slots in which entries can be made. This is accomplished by making lost+found, copying a number of files to the directory, and then removing them (before *fsck* is executed).

Checking the raw device is almost always faster.

FILES

/etc/fstab contains default list of file systems to check.

DIAGNOSTICS

The diagnostics produced by *fsck* are intended to be self-explanatory.

NAME

getty – set typewriter mode

SYNOPSIS

/etc/getty [char]

DESCRIPTION

Getty is invoked by *init*(8) immediately after a typewriter is opened following a dial-up. It reads the user's login name and calls *login*(1) with the name as argument. While reading the name *getty* attempts to adapt the system to the speed and type of terminal being used.

Init calls *getty* with a single character argument taken from the *ttys*(5) file entry for the terminal line, which is now ignored. *Getty* looks for the terminal line name with the *ttyname* call and then searches through the *ttytype* data base. This data base has 6 entries for each terminal line of the following form.

TERM	device	in speed	out speed	login mode	shell mode
pt80	tty0	300	300	PA;RW;CR1	PA;EC;TX;C1
pv	tty10	9600	9600	PA;RW	PA;EC

The first entry is the *TERM* variable for the terminal type (see also *termcap (5)*) , the second entry is the device name of the terminal line without '/dev/', the third and fourth entries are the terminal receive and transmit speed, the fifth and sixth are the tty modes for the login command and the normal shell mode. The entries must be separated by a ' ' or tab character, the tty modes by a ';' character. The meaning of the modes is as follows (see also *tty (4)*):

B0	BS0	DA	ALLDELAY
B1	BS1	DC	CRDELAY
CB	CBREAK	DB	BSDELAY
CM	CRMOD	DN	NLDELAY
CR0	CR0	DT	TBDELAY
CR1	CR1	DV	VTDELAY
CR2	CR2	N0	NL0
CR3	CR3	N1	NL1
PA	ANYP	N2	NL2
PE	EVENP	N3	NL3
P0	ODDP	EC	ECHO
T0	TAB0	TD	TANDEM
T1	TAB1	TX	XTABS
T2	TAB2	LC	LCASE
F0	FF0	F1	FF1
RW	RAW	HU	HPCL

When *getty* writes the '<node name> login:' greeting message to your terminal, answer with your user login name.

The user's name is terminated by a new-line or carriage-return character. In the second case CRMOD mode is set (see *ioctl*(2)).

The name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is nonempty, the system is told to map any future upper-case characters into the corresponding lower-case characters (Then \A is mapped to upper-case A).

login is called with the user's name as argument.

FILES

/etc/ttys
/etc/ttytype
/etc/gettydefs

SEE ALSO

init(8), login(1), [illegible], ttys(5), [illegible], tty(1)